**EXPERIMENT NO: 1A**

**AIM:STUDY OF SYSTEM PROGRAMS / SYSTEM SOFTWARES.**

**THEORY:**

**=>What is System Software?**

System software is software designed to provide a platform for other software. Examples of system software include operating systems like mac OS, Linux OS and Microsoft Windows, computational science software, game engines, industrial automation, and software as a service applications.

In contrast to system software, software that allows users to do user-oriented tasks such as create text documents, play games, listen to music, or browse the web are collectively referred to as application software. In the early days of computing most application software was custom-written by computer users to fit their specific hardware and requirements. In contrast, system software was usually supplied by the manufacturer of the computer hardware and was intended to be used by most or all users of that system.

**=>ASSEMBLERS:**

An assembler is a program that converts assembly language into machine code. It takes the basic commands and operations from assembly code and converts them into binary code that can be recognized by a specific type of processor.

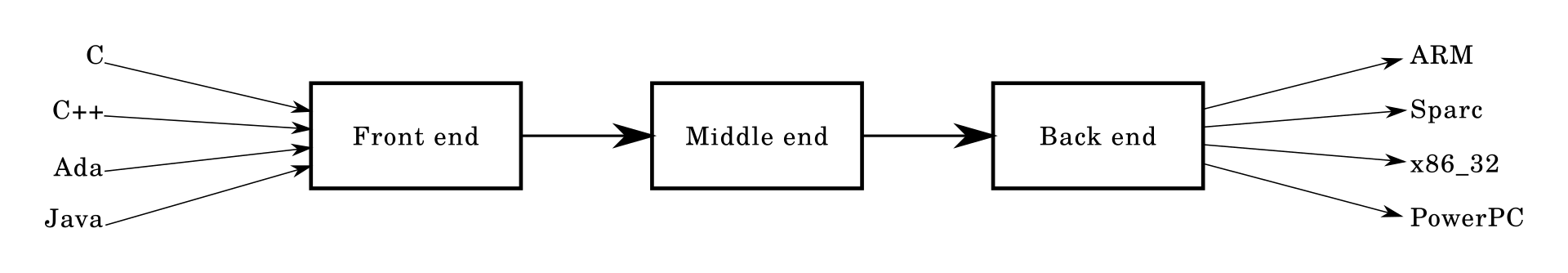
****

**Figure 1: Assembler working**

Assemblers are similar to compilers in that they produce executable code. However, assemblers are more simplistic since they only convert low-level code (assembly language) to machine code. Since each assembly language is designed for a specific processor, assembling a program is performed using a simple one-to-one mapping from assembly code to machine code. Compilers, on the other hand, must convert generic high-level source code into machine code for a specific processor.

**=>COMPILERS:**

A compiler is a computer program that translates computer code written in one programming language (the source language) into another language (the target language). The name compiler is primarily used for programs that translate source code from a high-level programming language to a lower level language (e.g., assembly language, object code, or machine code) to create an executable program.



**Figure 2: Three Stage Compiler Structure**

A compiler is likely to perform many or all of the following operations: preprocessing, lexical analysis, parsing, semantic analysis (syntax-directed translation), conversion of input programs to an intermediate representation, code optimization and code generation. Compilers implement these operations in phases that promote efficient design and correct transformations of source input to target output. Program faults caused by incorrect compiler behavior can be very difficult to track down and work around; therefore, compiler implementers invest significant effort to ensure compiler correctness.

**=>DEBUGGERS:**

A debugger or debugging tool is a computer program used to test and debug other programs (the "target" program). The main use of a debugger is to run the target program under controlled conditions that permit the programmer to track its operations in progress and monitor changes in computer resources (most often memory areas used by the target program or the computer's operating system) that may indicate malfunctioning code. Typical debugging facilities include the ability to run or halt the target program at specific points, display the contents of memory, CPU registers or storage devices (such as disk drives), and modify memory or register contents in order to enter selected test data that might be a cause of faulty program execution.

There are two types of debuggers:

1. CorDBG(command line debugger):in this, compilation of original file using debug switch is must.

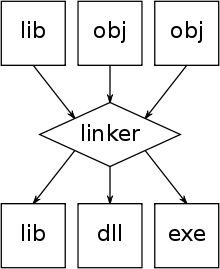
2. DbgCLR(graphic debugger):used by Visual Studio .NET

Some widely used debuggers are:

* Arm DTT, formerly known as Allinea DDT
* Eclipse debugger API used in a range of IDEs: Eclipse IDE (Java) Nodeclipse (JavaScript)
* Firefox JavaScript debugger
* GDB - the GNU debugger

**=>LINKERS:**

In computer science, a linker is a computer program that takes one or more object files generated by a compiler and combines them into one, executable program. Computer programs are usually made up of multiple modules that span separate object files, each being a compiled computer program. The program as a whole refers to these separately compiled object files using symbols. The linker combines these separate files into a single, unified program, resolving the symbolic references as it goes along.



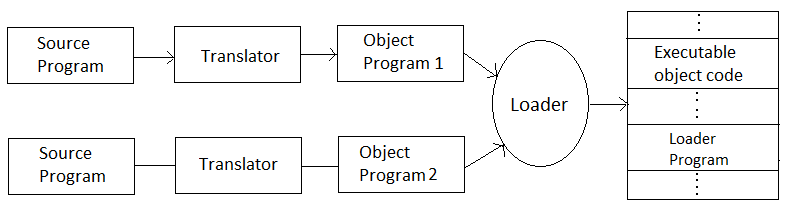
**Figure 3: Linker**

Dynamic linking is a similar process available on many operating systems, which postpones the resolution of some symbols until the program is executed. When the program is run, these dynamic link libraries are loaded, as well. Dynamic linking does not require a linker.

The linker bundled with most Linux systems is called ld. On Unix-like operating systems, ld is a linker. It combines a number of compiled object and archive files, relocates their data, and ties up symbol references. Usually, the last step in compiling a program is to run ld.

**=>LOADERS:**

In computer systems a loader is the part of an operating system that is responsible for loading programs and libraries. It is one of the essential stages in the process of starting a program, as it places programs into memory and prepares them for execution. Loading a program involves reading the contents of the executable file containing the program instructions into memory, and then carrying out other required preparatory tasks to prepare the executable for running. Once loading is complete, the operating system starts the program by passing control to the loaded program code.



**Figure 4: Loader Working**

All operating systems that support program loading have loaders, apart from highly specialized computer systems that only have a fixed set of specialized programs. Embedded systems typically do not have loaders, and instead, the code executes directly from ROM. In order to load the operating system itself, as part of booting, a specialized boot loader is used. In many operating systems, the loader resides permanently in memory, though some operating systems that support virtual memory may allow the loader to be located in a region of memory that is pageable.

**=>EDITORS:**

Editors or text editors are software programs that enable the user to create and edit text files. In the field of programming, the term editor usually refers to source code editors that include many special features for writing and editing code. Notepad, Wordpad are some of the common editors used on Windows OS and vi, emacs, Jed, pico are the editors on UNIX OS. Features normally associated with text editors are — moving the cursor, deleting, replacing, pasting, finding, finding and replacing, saving etc.

Types of Editors:

There are generally five types of editors as described below:

1.Line editor: In this, you can only edit one line at a time or an integral number of lines. You cannot have a free-flowing sequence of characters. It will take care of only one line.

Ex: Teleprinter, edlin, teco

2.Stream editors: In this type of editors, the file is treated as continuous flow or sequence of characters instead of line numbers, which means here you can type paragraphs.

Ex: Sed editor in UNIX

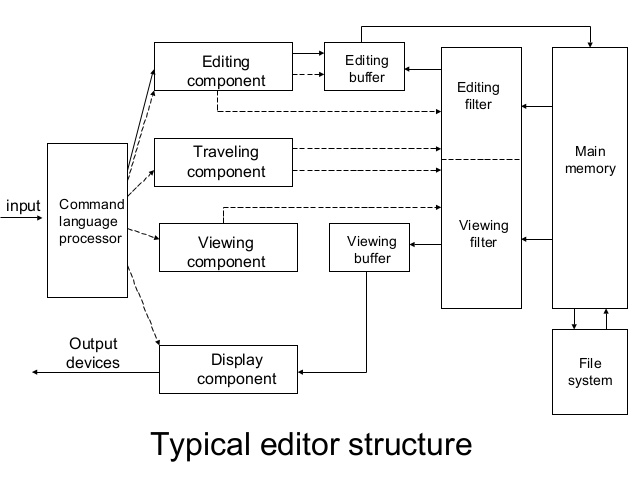
3.Screen editors: In this type of editors, the user is able to see the cursor on the screen and can make a copy, cut, paste operation easily. It is very easy to use mouse pointer.

Ex: vi, emacs, Notepad

4.Word Processor: Overcoming the limitations of screen editors, it allows one to use some format to insert images, files, videos, use font, size, style features. It majorly focuses on Natural language.

5.Structure Editor: Structure editor focuses on programming languages. It provides features to write and edit source code.

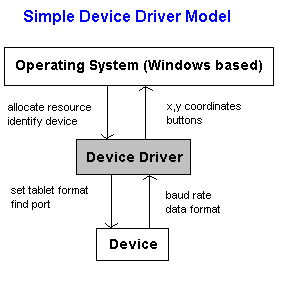
Ex: Netbeans IDE, gEdit.



**Figure 5: Typical Editor Structure**

**=>DEVICE DRIVERS:**

A device driver is a software program that controls a particular type of hardware device that is attached to a computer. When buying an operating system, many device drivers are built into the product. However, if a user later buys a new type of device that the operating system did not anticipate, the new device driver will have to be installed. A device driver essentially allows smooth communication between a connected hardware device and the operating system (OS).



**Figure 6: Simple Device Driver Structure**

Device drivers, particularly on modern Microsoft Windows platforms, can run in kernel-mode (Ring 0 on x86 CPUs) or in user-mode (Ring 3 on x86 CPUs).The primary benefit of running a driver in user mode is improved stability, since a poorly written user-mode device driver cannot crash the system by overwriting kernel memory. On the other hand, user/kernel-mode transitions usually impose a considerable performance overhead, thus making kernel-mode drivers preferred for low-latency networking.

**=>UTILITIES:**

Utility software is software designed to help to analyze, configure, optimize or maintain a computer. It is used to support the computer infrastructure - in contrast to application software, which is aimed at directly performing tasks that benefit ordinary users. However, utilities often form part of application systems. For example a batch job may run user-written code to update a database and may then include a step that runs a utility to back up the database, or a job may run a utility to compress a disk before copying files.

Although a basic set of utility programs is usually distributed with an operating system (OS), and this first party utility software is often considered part of the operating system, users often install replacements or additional utilities. Those utilities may provide additional facilities to carry out tasks that are beyond the capabilities of the operating system.

Many utilities that might affect the entire computer system require the user to have elevated privileges, while others that operate only on the user's data do not.

1.System utilities

Anti-virus, Clipboard managers, Computer access control software , Debuggers ,Diagnostic programs Network utilities ,Package managers ,Registry cleaners, System monitors ,System profilers

2.Storage device management utilities

Backup software,Disk checkers ,Disk compression utilities ,Disk defragmenters Disk formatters, Disk partition editors ,Disk space analyzers ,Tape initializers.

3.File management utilities

Archivers , Cryptographic utilities, Data compression utilities, Data conversion utilities, Data recovery utilities, Data synchronization utilities, File synchronization utilities, Revision control utilities, Disk cleaners, File comparison utilities, File managers.

4.Miscellaneous utilities

Data generators (e.g. IEBDG), Hex editors, HTML checkers, Installation or setup utilities. There are also Uninstallers, Patching utilities, Screensavers, Sort/Merge programs, Standalone macro .